Direct Multichannel Tracking

Carlos Jaramillo*
Computer Science Dept., CUNY Graduate Center
365 Fifth Avenue, New York, NY, USA
omnistereo@gmail.com

Yuichi Taguchi and Chen Feng
Mitsubishi Electric Research Labs (MERL)
201 Broadway, Cambridge, MA, USA
{taguchi,cfeng}@merl.com

Abstract

We present direct multichannel tracking, an algorithm for tracking the pose of a monocular camera (visual odometry) using high-dimensional features in a direct image alignment framework. Instead of using a single grayscale channel and assuming intensity constancy as in existing approaches, we extract multichannel features at each pixel from each image and assume feature constancy among consecutive images. High-dimensional features are more discriminative and robust to noise and image variations than intensities, enabling more accurate camera tracking. We demonstrate our claim using conventional hand-crafted features such as SIFT as well as more recent features extracted from convolutional neural networks (CNNs) such as Siamese and AlexNet networks. We evaluate the performance of our algorithm against the baseline case (single-channel tracking) using several public datasets, where the AlexNet feature provides the best pose estimation results.

1. Introduction

Visual odometry and simultaneous localization and mapping (SLAM) are an important building block for several applications, such as robot navigation, autonomous driving, and augmented reality. The majority of conventional approaches are based on sparse feature extraction and matching [6, 18]: They extract sparse features from each image, match them among images with RANSAC, and triangulate correspondences to obtain 3D landmarks, which are further matched with the subsequent images. Recently, *direct* methods that do not require sparse feature extraction have emerged [26, 10, 8, 12]: They are based on direct image alignment that matches dense or semi-dense pixels between images assuming intensity constancy. Although these methods demonstrated impressive results, raw intensity values are not discriminative and matching them has difficulties in weakly textured scenes and due to big illumination changes.

In this paper, we propose direct multichannel tracking, which extends the existing direct method using a single intensity channel by using multichannel features extracted at each pixel. We use conventional hand-crafted features such as SIFT [21] and features extracted from convolutional neural networks (CNNs) [19, 33]. These features are computed on a patch around each pixel and contain more context than the single pixel. Thus, they are more discriminative than intensities and can match even with large image variations. Moreover, direct methods mainly use pixels with large gradients, i.e., edges, as valid 3D landmarks. The number of edges is limited in single-channel intensity images, thus existing approaches have difficulties in weakly textured scenes. On the other hand, the multichannel features can have gradients in a wider variety of regions (e.g., blob regions for SIFT, regions that match the convolutional filter shape for CNNs), increasing the number of valid 3D landmarks and leading to better tracking accuracy. We demonstrate these advantages of our algorithm in experiments using several public benchmark datasets [13, 28, 16, 8, 11].

1.1. Related work

1.1.1 Visual odometry and SLAM

MonoSLAM [6] and PTAM [18] were earlier examples of real-time SLAM systems based on the sparse feature extraction and matching. Those approaches have been sophisticated and demonstrated state-of-the-art performance in several practical scenarios [25, 24]. Direct methods [26, 10, 12] have been proposed as an alternative to the sparse-feature-based approaches. They do not extract features, but rather directly use raw intensity values and minimize the photometric errors of corresponding pixels for estimating depth maps and camera poses. Newcombe *et al*. [26] used an entire set of pixels and implemented a real-time camera tracking and dense depth reconstruction system on GPU. Engel *et al*. [10] showed that using semi-dense pixels that have non-negligible gradients (i.e., edges) is sufficient for accurate camera tracking in real-time. Forster *et al*. [12] used patches around sparse keypoints with direct alignment to estimate an initial camera pose, which was further refined

*This work was done while interning at MERL.

using sparse-feature-based bundle adjustments.

Engel *et al*. [8] demonstrated large-scale direct SLAM (LSD-SLAM), which incorporated the semi-dense visual odometry algorithm [10] into a SLAM framework with keyframe selection and global pose graph optimization to reconstruct large-scale 3D models. Engel *et al*. [7] also extended [10] by using window-based bundle adjustment that jointly minimizes the photometric errors in multiple consecutive images. A multi-level mapping approach to LSD-SLAM was proposed by Greene *et al*. [14] such that correspondence search can take place at coarser resolutions in order to fill up holes in the depth map and improve the tracking accuracy. Note that those extensions are orthogonal to our extension using multichannel features: Our algorithm can be easily incorporated into the LSD-SLAM framework [8] with global pose graph optimization and window-based bundle adjustment [7] and multi-level mapping [14]. Hence, in this paper, we evaluate our algorithm using multichannel features with respect to the baseline algorithm using the single-channel intensities [10].

As concurrent work, Park *et al*. [27] evaluated the robustness of direct camera tracking under illumination changes with descriptor fields [5] which separate the intensity gradients (computed via a kernel) into 4 channels based on their respective signs. They also evaluated simple scalar local descriptors such as the gradient magnitude and the more sophisticated census transform [32], both of which were observed to perform well for the real-world dataset [30] evaluated upon. Alismail *et al*. [1, 2] presented the use of Bit-Planes as a binary multichannel feature for direct visual odometry. The Bit-Planes feature is a simple 8-bit binary feature descriptor extracted on a small patch of 3×3 pixels by comparing the central pixel to its 8 surrounding neighbors via a boolean operator such as "$>$". Their paper mainly focused on feature description speed and lighting robustness, while our paper shows that the use of multichannel features can improve tracking accuracy for general real-world sequences. In fact, we also provide direct comparisons between the Bit-Planes feature and our features.

1.1.2 Feature-based dense correspondences

SIFT flow [20] was a pioneering work for using multichannel features instead of intensities to obtain dense correspondences between images. They extracted SIFT features at each pixel from each image and used the feature distances as the data term in a discrete optimization framework to obtain dense optical flow fields. Features extracted from convolutional neural networks (CNNs) have been recently used in such a framework instead of the conventional hand-crafted features for stereo [34, 33, 4, 23] and optical flow [15] problems. Those CNNs, commonly referred to as Siamese networks, generate a high-dimensional feature vector from each image patch, and are trained such that similar image patches produce similar features. The features are matched with each other using the L_2 distance or another neural network, and the matching distances are fed into a discrete optimization framework to obtain smooth results. Note that these existing approaches explicitly match the features by searching for the minimum feature distance. In contrast, the tracking part of our algorithm is based on a differential method that implicitly finds the matching features based on their gradient, which is rooted in the seminal Lucas-Kanade optical flow algorithm [22]. Moreover, to the best of our knowledge, our paper is the first attempt to use pixel-wise CNN features for the visual odometry problem.

2. Direct multichannel tracking

This section describes the proposed direct multichannel tracking algorithm and the multichannel features used with it. Our method is built upon the semi-dense visual odometry algorithm [10] and implemented from the source code publicly released as a part of the LSD-SLAM system [8]. We thus follow their notation and mainly describe our extensions that replace the single intensity channel with multichannel features.

2.1. Multichannel feature extraction

Our algorithm first extracts an N-dimensional feature $\boldsymbol{F}(\mathbf{p}_i)$ for each pixel location $\mathbf{p}_i$ from the input single-channel intensity image I or 3-channel color image $\boldsymbol{C}$. Such features are more discriminative than the original intensity or color of the pixel alone since they are computed on a patch around the pixel and include more context from the surrounding pixels. In this paper, we use SIFT [21] as a conventional hand-crafted feature, in addition to learning-based features obtained from two pre-trained CNNs [19, 33].

2.1.1 Dense SIFT feature

SIFT has been considered one of the most robust features under several image variations including scale, in-plane rotation, and illumination changes. Liu *et al*. [20] exploited it to compute dense optical flow fields under large image variations. We follow their approach and obtain pixel-wise SIFT descriptors. Specifically, we adapted the code from [20] to compute the 128-dimensional SIFT descriptors at a single scale (original image resolution) and without computing the dominant orientation direction.

2.1.2 CNN features

With the recent progress of deep learning, data-driven feature extraction based on CNNs are shown to have better performance than hand-crafted ones, in both high-level (object

detection, classification, etc.) and low-level (stereo matching, optical flow, etc.) computer vision tasks. Instead of using a fixed feature transformation such as SIFT, learning-based methods search among a large functional space of feature transformations for one that generalizes well to different cases using a large amount of training data.

One type of CNNs, such as the above-mentioned Siamese networks [34, 33, 4, 23, 15], was designed and trained specifically for the low-level vision task of patch matching: They take an image patch as the input in order to output a high-dimensional feature vector suitable for matching. These CNNs follow the same design concept as SIFT, describing a pixel on the image with its neighborhood information using a fixed-dimensional vector, i.e., a feature descriptor. Meanwhile, another type of CNNs, such as AlexNet [19], was designed and trained for high-level vision tasks over a whole image. It has been shown that the respective layers' outputs (i.e., features) form a hierarchy, where low layers correspond to corners and edges, middle layers to textures, and high layers to class-specific patches [35]. In particular, the low layers use convolutions over local image patches and act as a filter bank, from which we extract low-level features.

We used two representative CNNs among those two types: the "siam" network from [33] (which we refer to as Siamese from here on) and AlexNet [19]. We used publicly available pre-trained weights with certain modifications to properly apply them to our problem:

Siamese feature: The original Siamese network was designed to work with single-channel (grayscale) image patches of 64×64 pixels as the input, and a 256-dimensional feature vector as the output for each image patch. We forward propagated the 64×64 patch centered around each pixel location of the input image, and then composed all the outputs into an $H \times W \times 256$ feature map.

AlexNet feature: We used the first convolution layer "conv1" followed by a rectified linear unit (ReLU) to obtain an $H \times W \times 96$ feature map. To make the output resolution the same as the input resolution, we changed (1) the original input resolution of 227×227 pixels (for 3 color channels) to $H \times W$, and (2) for "conv1," changed the original stride of 4 to 1 and used zero-padding of size 5, while maintaining the kernel size of 11. Note that these modifications do not affect the learned convolution kernel weights.

2.2. Camera pose tracking

In the baseline single-channel algorithm [10, 8], the pose of the current image I with respect to a reference image I_{ref} is estimated by assuming the intensity constancy and minimizing a photometric error between the two images. Specifically, given I_{ref} and its inverse depth map estimate D_{ref}, the photometric error for $V(\leq HW)$ points is

$$E(\boldsymbol{\xi}) = \sum_{i=1}^{V} \Big(\underbrace{I_{\text{ref}}(\mathbf{p}_i) - I(\omega(\mathbf{p}_i, D_{\text{ref}}(\mathbf{p}_i), \boldsymbol{\xi}))}_{:=r_i(\boldsymbol{\xi})} \Big)^2, \quad (1)$$

where $\boldsymbol{\xi}$ is a 6-vector representing the pose of the current image I with respect to the reference image I_{ref} in Lie algebra $\mathfrak{se}(3)$, and ω is the 3D projective warp function that maps the pixel location $\mathbf{p}_i$ in the reference image according to its inverse depth $D_{\text{ref}}(\mathbf{p}_i)$ and the pose $\boldsymbol{\xi}$ to the pixel location in the current image. Starting from an initial pose estimate $\boldsymbol{\xi}^{(0)}$, the pose $\boldsymbol{\xi}$ is iteratively updated via the pose concatenation operator $\circ$ such that

$$\boldsymbol{\xi}^{(n+1)} = \delta\boldsymbol{\xi}^{(n)} \circ \boldsymbol{\xi}^{(n)}, \quad (2)$$

where the incremental update $\delta\boldsymbol{\xi}^{(n)}$ at the n^{th} iteration is estimated using the Levenberg-Marquardt (LM) method

$$\delta\boldsymbol{\xi}^{(n)} = -\mathbf{A}^{-1}\mathbf{J}^{\mathsf{T}}\mathbf{r}, \quad \mathbf{A} = \mathbf{J}^{\mathsf{T}}\mathbf{J}, \quad \mathbf{J} = \left.\frac{\partial \mathbf{r}(\boldsymbol{\epsilon} \circ \boldsymbol{\xi}^{(n)})}{\partial \boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=0}. \quad (3)$$

Here, $\mathbf{J}$ is the Jacobian matrix computed for the stacked residual vector $\mathbf{r} = (r_1, \ldots, r_V)^{\mathsf{T}}$ using the current pose estimate $\boldsymbol{\xi}^{(n)}$. The diagonal components of $\mathbf{A}$ are multiplicatively adjusted by a $1 + \lambda_{LM}$ factor.

In our algorithm, we assume feature constancy, instead of intensity constancy, and replace the photometric error (1) with the feature-based error

$$E(\boldsymbol{\xi}) = \sum_{i=1}^{V} \Big\| \underbrace{\boldsymbol{F}_{\text{ref}}(\mathbf{p}_i) - \boldsymbol{F}(\omega(\mathbf{p}_i, D_{\text{ref}}(\mathbf{p}_i), \boldsymbol{\xi}))}_{:=\mathbf{r}_i(\boldsymbol{\xi})} \Big\|^2, \quad (4)$$

where $\boldsymbol{F}$ and $\boldsymbol{F}_{\text{ref}}$ are the N-dimensional feature maps for the current and reference images, respectively. Note that each residual $\mathbf{r}_i$ is now an N-dimensional vector (instead of a single value r_i), and thus we use the squared L_2 norm to compute a per-pixel error. The feature-based error (4) can be minimized iteratively over the same equation (3) with the new stacked residual vector $\mathbf{r} = ({\mathbf{r}_1}^{\mathsf{T}}, \ldots, {\mathbf{r}_V}^{\mathsf{T}})^{\mathsf{T}}$, which is N times larger than the original one. $\mathbf{J}$ can be constructed by stacking N Jacobian matrices of size $V \times 6$ computed independently for each of the N channels. Note that the entire $(NV) \times 6$ matrix $\mathbf{J}$ is never stored, but rather each row of the linear system is accumulated in the 6×6 matrix $\mathbf{A}$ and the 6-vector $\mathbf{J}^{\mathsf{T}}\mathbf{r}(\boldsymbol{\xi}^{(n)})$ on the fly. Thus this multichannel extension does not increase the memory size requirement; the computational cost still increases N times, but the computation can be easily parallelized.

In practice, the baseline algorithm [10, 8] evaluates the photometric error (1) only on a semi-dense set of V points whose pixel projections lie within the image bounds and whose gradient magnitudes are larger than certain threshold

g_{th}. For each of the valid pixels, the gradient magnitude and the inverse depth variance estimate [17, 10] are used as weight criteria, and the Huber norm is used as a robust metric under the presence of noise and outliers [8]. In our multichannel algorithm, we compute a single gradient magnitude by averaging over the N channels. Except for this extension and the use of the feature-based error (4), our algorithm is implemented upon the same techniques as for the baseline single-channel case.

2.3. Inverse depth map estimation

The baseline algorithm uses the latest keyframe selected in the SLAM framework [8] as the reference image and maintains the inverse depth map estimate on the reference image. Given the pose estimate of the current image with respect to the reference image, the inverse depth map estimate is updated as follows. For each of the valid pixels (existent or newly-observed) in the reference image, the epipolar line is computed on the current image and then a correspondence search is performed along the epipolar line, whose search interval is limited by the current variance estimate of the inverse depth of the reference pixel. In the single channel case, the sum-of-squared-differences (SSD) error over five equidistant neighboring points along the epipolar line is deemed as the matching cost. Then, the inverse depth is updated according to the best matching pixel location with sub-pixel interpolation. In our algorithm, we replace the SSD error between the intensities with the squared L_2 norm error between the N-dimensional features. Again, except for this extension, we use the same set of techniques employed in the baseline single-channel implementation.

2.4. Feature scaling

Generally the values of the N-dimensional feature vectors can be distributed in any range (they might be normalized to $[0, 1]$, or they might take arbitrary real values), which is implementation dependent. Theoretically this is not a problem, because only relative values are important for computing the feature distances and gradients in the camera pose tracking and inverse depth estimation. However, in practice, several parameters are set by assuming a predetermined range of values (e.g., $[0, 255]$ for the single intensity channel of the baseline algorithm). In particular, we found that the gradient magnitude threshold, g_{th}, used to determine the set of V valid pixels and the density of the inverse depth map as a consequence, is an influential parameter, which depends on the range of the values.

Instead of tuning such parameters for each feature, we normalized each feature with a scale factor obtained as follows. Our goal was to use all the default parameters set by the authors of the baseline algorithm [10, 8] for the single intensity channel ranging between $[0, 255]$. Simply normalizing the maximum feature value to become 255 did not perform well, because the distributions of the intensity values and feature values were different and such a normalization made the feature gradients of too many pixels close to zero. We instead used an approach that matches the average number of the semi-dense pixels having gradient magnitudes larger than the threshold g_{th} between the single-channel intensity and the N-channel features. In order to achieve fair quantitative comparisons during our experiments, we obtained a single global scale factor for each multichannel feature (SIFT, Siamese, and AlexNet) by computing the average number of valid pixels across several images sampled throughout the datasets as a preprocessing step. We refer to the scale factor obtained this way as the "default feature scale" since it maintains the original parameters established by the authors of the baseline algorithm [10, 8] regardless of the feature employed in the experiment.

3. Experiments

We performed experiments on five publicly available datasets of monocular image sequences summarized in Table 1. We evaluated the pose estimation accuracy for our algorithm using different multichannel features, including SIFT, Siamese, and AlexNet, for comparison against the baseline algorithm that only uses the grayscale intensity. For completeness, we also evaluated (1) the "Affine Lighting Model" for grayscale (denoted as Gray-A here) that was described in [9] but used as implemented in the source code accompanying [8], and (2) the Bit-Planes feature [1, 2], both of which were designed to be robust to illumination variations. With the KITTI, ICL-NUIM, and Tsukuba datasets, we also tested tracking under RGB and Lab color spaces, acting as trivial multichannel features.

To obtain consistent results, we removed randomness from the original LSD-SLAM implementation by (1) disabling the threading mechanisms and (2) using the same set of random values for different methods to initialize the inverse depth map D at $t = 0$. As described in Section 2.4, we determined the default feature scales globally during a preprocessing step, and we used the same set of parameters from the original LSD-SLAM implementation [8]. Feature scales for Bit-Planes, SIFT, Siamese, and AlexNet were 9.77, 0.68, 347.50, and 1.23, respectively.

3.1. Evaluation criteria

The relative pose error (RPE) [30, 13] and the absolute trajectory error (ATE) [30, 16] have been common evaluation metrics for visual odometry algorithms using stereo or RGB-D cameras, where the pose can be computed without the scale ambiguity. However, due to the inherent scale ambiguity of monocular visual odometry, these metrics cannot be directly used and require some modifications. ATE, which is based on the global registration between the entire ground truth and estimated trajectories, is hard to assess

Dataset Name	# Seqs	Color	Res.	FPS	Type (Ground Truth)	Environment	Other characteristics
TUM DSO [11]	50	Gray	SXGA	50	Real (VICON)	In/Out-doors	Hand-held motion, scene and illumination variety.
KITTI Odometry [13]	11	RGB	∼WXGA	10	Real (GPS+IMU)	Outdoors (roads)	Fast motion, low frame rate, long paths
ICL-NUIM [16]	8	RGB	VGA	30	Synthetic (Yes)	Indoors	Photorealistic, short paths
Tsukuba (New) [28]	4	RGB	VGA	30	Synthetic (Yes)	Indoors	Photorealistic, single path, quick turns, various illumination modes
LSD-SLAM [8]	4	Gray	VGA	50	Real (None)	In/Out-doors	Hand-held motion, short and long trajectories

Table 1. Datasets. The first 4 were used for quantitative evaluation. LSD-SLAM was only used for 3D reconstruction comparison.

with because of the scale ambiguity; even if we estimate the scale with a similarity transformation between the trajectories, there is the scale drift problem [29, 8] that is difficult to solve in pure visual odometry algorithms without a SLAM framework. In fact, ATE has a bias toward shorter tracking, as it tends to be low if the number of tracked frames (one of our evaluation criteria) is small. RPE does not have the bias and can also measure the drift.

We thus used slightly modified versions of RPE for the quantitative evaluation. Given a sequence of poses as the transformation matrices from the estimated trajectory $\mathbf{P}_1, \ldots, \mathbf{P}_T$ and that from the ground truth trajectory $\mathbf{Q}_1, \ldots, \mathbf{Q}_T$, the relative poses between a pair of images (i, j) can be obtained as $\mathbf{P}_{ij} = \mathbf{P}_i^{-1}\mathbf{P}_j$ and $\mathbf{Q}_{ij} = \mathbf{Q}_i^{-1}\mathbf{Q}_j$ for the estimated and ground truth trajectories, respectively. We compute two types of error metrics between the pair of images (i, j) as follows:

RPE rotation error: For the error in the rotation component, we first obtain the difference between the rotation matrices as $\Delta\mathbf{R}_{ij} = R(\mathbf{Q}_{ij})^\mathsf{T} R(\mathbf{P}_{ij})$, where $R(\cdot)$ takes the rotation matrix of the transformation matrix. We then compute the rotation error as the angle of $\Delta\mathbf{R}_{ij}$:

$$\theta_{\Delta\mathbf{R}_{ij}} = \arccos\left(\frac{(\mathrm{Tr}\,(\Delta\mathbf{R}_{ij}) - 1)}{2}\right). \tag{5}$$

RPE translation angle error: For the error in the translation component, we compute the angle between the translation vectors taken by $t(\cdot)$ as follows:

$$\theta_{\Delta\boldsymbol{t}_{ij}} = \arccos\left(\frac{t(\mathbf{Q}_{ij})^\mathsf{T} t(\mathbf{P}_{ij})}{\|t(\mathbf{Q}_{ij})\|\|t(\mathbf{P}_{ij})\|}\right), \tag{6}$$

which is independent of the scale ambiguity.

For each sequence, we computed these error metrics for a set of 8 uniformly-divided path lengths as in [13]. Since we used different datasets, the longest path length was considered an arbitrary ratio r_l of the entire path length. We used $r_l = \frac{1}{3}$ for TUM DSO and ICL-NUIM, whereas $r_l = \frac{1}{20}$ for KITTI and Tsukuba. We then computed the average of the rotation and translation angle errors normalized by their respective path lengths as the overall errors, each of which was used to rank the feature's pose estimation accuracy according to the standard competition ranking ("1224"). We also ranked the features according to the number of frames successfully tracked because different features diverge at different frames and because some frames may not be registered with ground truth information.

Feature	Tracked Frames		Rotation Error		Trans. Ang. Error	
	Wins Count	Average Ranking	Wins Count	Average Ranking	Wins Count	Average Ranking
Gray	21	2.76	3	3.58	7	3.24
Gray-A	21	2.60	3	3.62	4	3.40
Bit-Planes	15	3.18	3	4.40	5	4.14
SIFT	20	2.82	12	3.18	6	3.72
Siamese	0	5.54	10	3.72	4	4.06
AlexNet	**37**	**1.74**	**19**	**2.50**	**24**	**2.44**

Table 2. Experimental rankings for the TUM DSO dataset.

Feature	Tracked Frames		Rotation Error		Trans. Ang. Error	
	Wins Count	Average Ranking	Wins Count	Average Ranking	Wins Count	Average Ranking
Gray	2	4.64	1	3.09	2	3.27
Gray-A	2	4.64	2	3.09	2	**2.36**
Bit-Planes	6	1.91	0	6.36	0	6.82
SIFT	**11**	**1.00**	0	6.73	0	7.36
Siamese	0	7.27	1	5.36	0	5.27
AlexNet	0	7.00	1	5.73	1	5.09
RGB	2	3.73	2	**2.73**	**3**	2.45
Lab	8	1.45	**4**	2.91	**3**	3.36

Table 3. Experimental rankings for the KITTI dataset.

Feature	Tracked Frames		Rotation Error		Trans. Ang. Error	
	Wins Count	Average Ranking	Wins Count	Average Ranking	Wins Count	Average Ranking
Gray	4	3.25	1	4.75	2	4.25
Gray-A	3	2.88	0	4.38	0	4.62
Bit-Planes	**5**	**2.62**	1	3.50	2	**3.00**
SIFT	4	4.38	**3**	**2.62**	**3**	3.12
Siamese	3	4.12	0	5.62	0	4.75
AlexNet	2	2.75	0	5.88	0	5.75
RGB	3	**2.62**	2	3.62	1	4.38
Lab	0	7.50	1	5.62	0	6.12

Table 4. Experimental rankings for the ICL-NUIM dataset.

Feature	Tracked Frames		Rotation Error		Trans. Ang. Error	
	Wins Count	Average Ranking	Wins Count	Average Ranking	Wins Count	Average Ranking
Gray	0	4.00	1	3.75	0	4.00
Gray-A	0	4.25	0	6.25	0	4.25
Bit-Planes	0	3.00	**3**	**2.00**	1	4.25
SIFT	**3**	**2.00**	0	5.00	0	6.00
Siamese	0	6.75	0	5.25	0	6.75
AlexNet	0	5.75	0	6.50	**2**	3.25
RGB	1	4.00	0	3.50	1	**3.00**
Lab	1	5.75	0	3.75	0	4.50

Table 5. Experimental rankings for the Tsukuba dataset.

3.2. Quantitative results

Due to space limitations, we only present a handful of experimental results that convey the claim of improved tracking accuracy by means of direct multichannel tracking. Figure 1 shows the average rotation error plots for the first 6 sequences of the TUM DSO dataset. The brackets in the legends specify the number of frames that were successfully tracked. All missing plots are given in the supplementary materials, including translation angle evaluations.

Tables 2, 3, 4, and 5 summarize the collected global met-

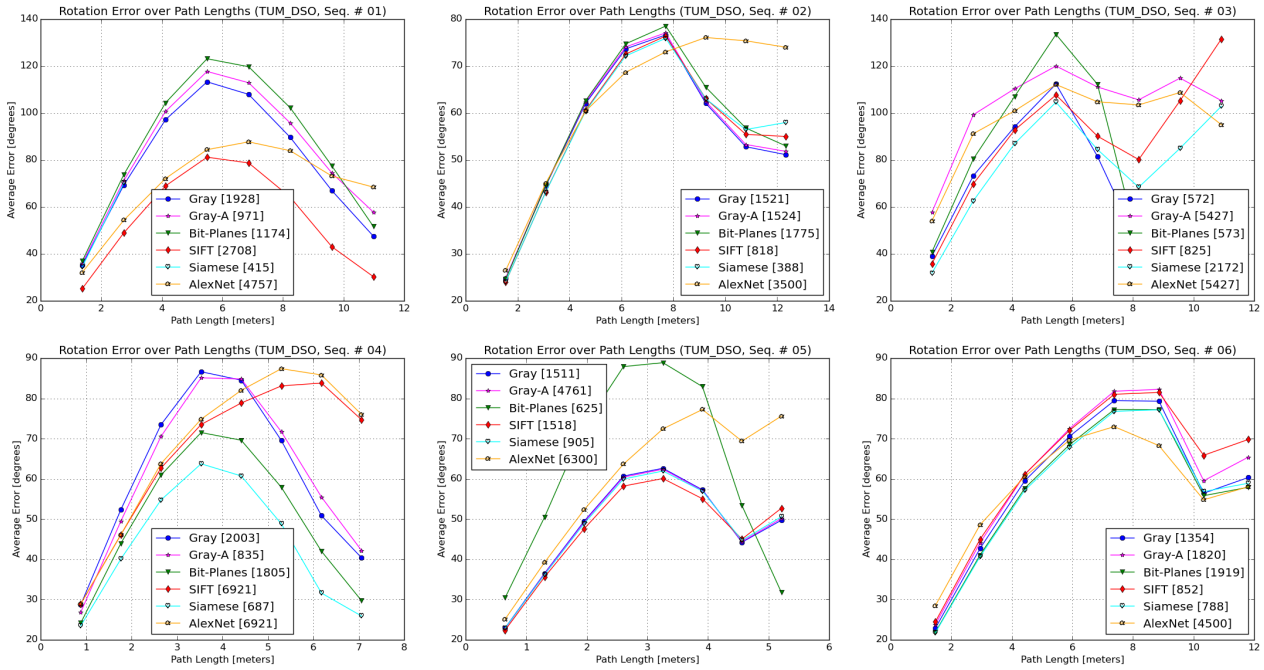
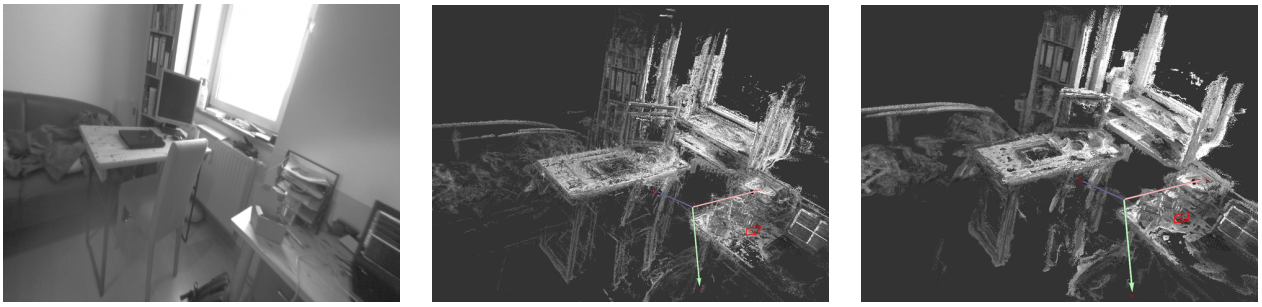


Figure 1. RPE rotation evaluation for the first 6 sequences of the TUM DSO dataset. In general, tracking of the single channel ("Gray") is less accurate than the proposed multichannel features.

Average Metric \ **Feature**	Gray	Gray-A	Bit-Planes	SIFT	Siamese	AlexNet
Number of Tracked Frames	2336	2604	2095	2361	636	**3131**
RPE Rotation Error $\left[\frac{\text{degree}}{\text{meter}}\right]$	7.44	7.31	13.73	8.17	8.10	**6.14**
RPE Trans. Ang. Error $\left[\frac{\text{degree}}{\text{meter}}\right]$	18.05	16.06	38.02	21.86	20.90	**12.93**
ATE [meters]	0.38	0.46	0.45	0.37	**0.07**	0.65

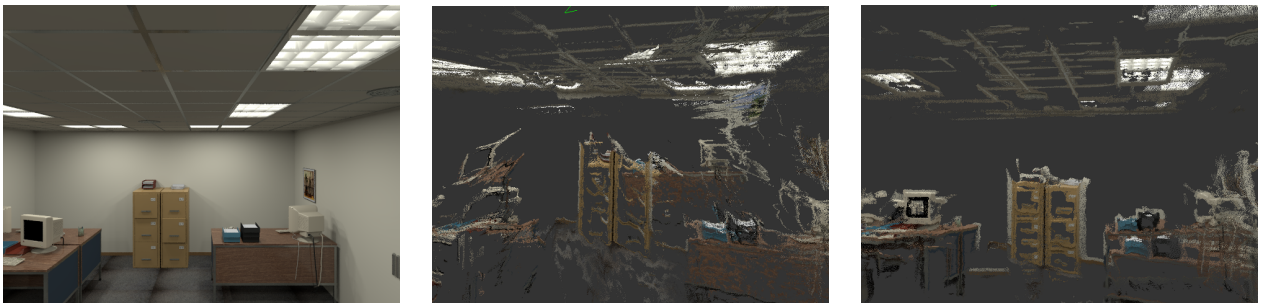
Table 6. Overall averages among the 50 sequences from the TUM DSO dataset.

rics for each dataset in terms of ranking. We counted the number of winning instances (first place rankings), and we also computed the ranking averages for every ranking criteria (number of tracked frames and pose estimation error metrics) of each participating feature across all sequences in that given dataset. All sequence-wise average errors associated with these rankings are included in the supplementary materials. Since TUM DSO is the largest dataset we evaluated upon, Table 6 contains its overall averages due to number of tracked frames and RPE's normalized components, where the accuracy trend of the proposed multichannel features can be verified. From these plots and tables, it is clear that the dense SIFT feature can succesfully track long trajectories. On the other hand, AlexNet and Siamese features are capable of improving the pose estimation accuracy over the baseline method. In particular, the AlexNet feature was a clear winner for the TUM DSO dataset, achieving smaller errors and tracking larger numbers of frames (fewer tracking failures) on average when compared against the others. Nevertheless, the compared Bit-Planes have a tendency toward accurate tracking under challenging illumination conditions as confirmed by the experiments under the Tsukuba dataset (Table 5) although we see that AlexNet and RGB seem better according to the translation angle RPE metric. Interestingly, the Siamese feature (trained specifically for low-level patch matching) did not perform better than the AlexNet feature (trained for a high-level classification task). A possible explanation is that the receptive field of the Siamese network (64×64 pixels) was too large for the tracking task due to occlusions and non-fronto-parallel planes compared to that of AlexNet (11×11 pixels). The need for small neighborhood descriptors was also noticed in [1]. For the ICL-NUIM dataset (Table 4), the SIFT feature performed the best in terms of accuracy, but the Bit-Planes and baseline methods performed equally well. This is because the ICL-NUIM dataset is synthetic and the intensity constancy holds throughout the sequences. Although the evaluation rankings for the KITTI dataset (Table 3) seem inconclusive, we must notice that these sequences do not satisfy the motion continuity required for direct tracking due to the low camera rate and high vehicle speeds. Our experiments agreed that although the basic color spaces (RGB



(a) Intensity image at $t = 150$ (b) Point cloud using Gray (baseline case) (c) Point cloud using AlexNet

Figure 2. 3D point cloud comparison for the `Room` sequence from the LSD-SLAM dataset. Observe the higher accuracy achieved via the AlexNet features for reconstructing furniture, books, computer monitor, among other objects.



(a) Color image at $t = 0$ (b) Point cloud using Gray (baseline case) (c) Point cloud using AlexNet

Figure 3. 3D point cloud comparison for the `Office` Seq. N^{o} 00 from the ICL-NUIM dataset. AlexNet features produced a denser and more accurate reconstruction. For example, the computer desks, file cabinets, and ceiling regions are better aligned for the most part.

and Lab) could be more discriminative than the grayscale channel, they do not contribute much to tracking accuracy, because their support is still a single pixel, not a local image patch as it is the case for our multichannel features.

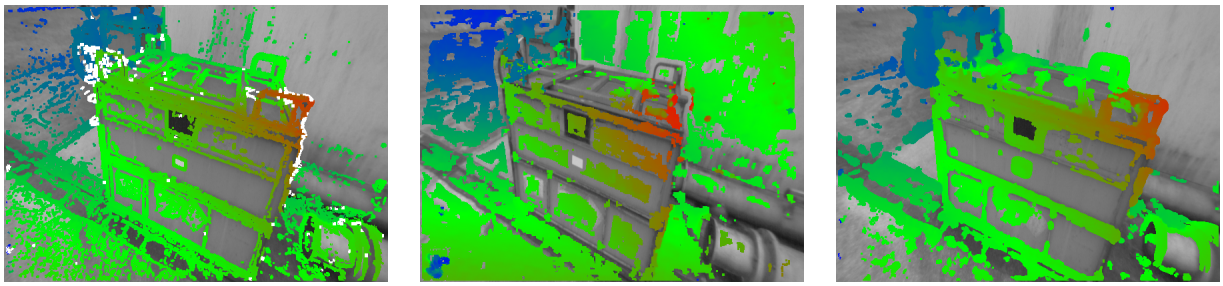
Despite the disadvantages of ATE as described above, we measured ATE as the average frame distance after aligning trajectories with the best 7-degree-of-freedom transformation (scale, rotation, and translation) obtained with [31]. For example, the overall ATE averages for TUM DSO are given in Table 6, and the complete results are given in the supplementary materials, where we also observe a trend for our multichannel methods outperforming grayscale. Although the Siamese feature appears to win, it registered much fewer frames, showing the aforementioned bias inherent in ATE.

3.3. Qualitative results

A qualitative advantage of employing high-dimensional features while tracking is the increased density of the reconstructed depth maps (Section 2.3). Figures 2 and 3 exemplify the improved 3D reconstruction accuracy due to tracking on AlexNet features in comparison to the baseline case. Observe how the multichannel features produced denser and more precise 3D point clouds in this couple of examples. In the supplementary materials, we include more results obtained from the ICL-NUIM dataset as well as for the `Room`, `Machine`, and `ECCV` sequences from the LSD-SLAM dataset. Even though the LSD-SLAM dataset did not provide positional ground truth data, we considered it for qualitative comparisons due to real-life long paths that appeared in the original LSD-SLAM manuscript [8]. Figure 4 compares the depth map activation $\mathbf{D}_{625}$ among feature cases (reference keyframes may not be the same) from the `Machine` sequence. We observe the number of negative depth points (white) is reduced for the multichannel cases.

3.4. Toward real-time implementation

A limitation of our approach is the reduced processing speed: Compared to the baseline single-channel case, multichannel feature generation followed by multichannel tracking requires an additional processing time. Table 7 (top) shows average processing times of the feature generation and tracking stages for different methods, measured on a standard PC with an Intel® Core™ i7-4790K CPU @ 4 GHz and an NVIDIA® GeForce® GTX 1080 GPU. Note that the feature generation stage also accounts for file I/O, undistortion, and cropping times. Out of our proposed features, AlexNet provides the fastest speed of $\sim$ 2 fps, yet slower than the baseline single-channel case at $\sim$ 33 fps.



(a) Gray (baseline case) (b) SIFT (c) AlexNet

Figure 4. Depth maps for the `Machine` sequence at $t = 625$ (negative depth shown in white).

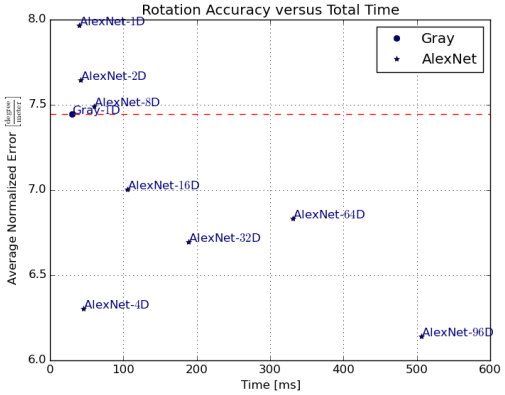


Figure 5. Performance analysis for the AlexNet features after dimensionality reduction. The total time is significantly reduced while the higher accuracy is maintained (for most instances) when compared against the grayscale baseline tracking algorithm.

	Feature	Generation (ms)	Tracking (ms)	Total (ms)	Total (fps)
	Gray (1D)	1.0×10^1	2.0×10^1	30	33
	Gray-A (1D)	1.0×10^1	2.1×10^1	31	32
	Bit-Planes (8D)	2.0×10^1	1.2×10^2	140	7
	SIFT (128D)	5.3×10^2	7.4×10^2	1270	1
	Siamese (256D)	8.0×10^3	1.0×10^3	9000	$\frac{1}{9}$
	AlexNet (96D)	7.9×10^1	4.3×10^2	509	2
Reduced	AlexNet-64D	6.1×10^1	2.7×10^2	331	3
Reduced	AlexNet-32D	4.0×10^1	1.5×10^2	190	5
Reduced	AlexNet-16D	2.9×10^1	7.6×10^1	105	10
Reduced	AlexNet-8D	2.4×10^1	3.6×10^1	60	17
Reduced	AlexNet-4D	2.2×10^1	2.3×10^1	45	22
Reduced	AlexNet-2D	2.1×10^1	2.1×10^1	42	24
Reduced	AlexNet-1D	2.0×10^1	2.0×10^1	40	25

Table 7. Average times sampled from the TUM DSO dataset.

Toward real-time direct multichannel tracking, we implemented a simple solution by reducing the original 96-dimensional AlexNet feature to K dimensions using principal component analysis (PCA). Similar to the computation of the default feature scale (Section 2.4), we computed the AlexNet features on several images sampled throughout the dataset and obtained PCA basis vectors as a preprocessing step. For online processing, we implemented the linear transformation using the basis vectors as a 3D convolution with a 1×1 kernel at the end of the AlexNet feature extraction on the GPU. Table 7 (bottom) shows the results using different number of dimensions K. Note that the feature generation time decreases as K decreases, despite the fact that the feature reduction step is an additional process to the original AlexNet feature generation. This indicates that the majority (about $3/4$) of the original AlexNet feature generation time was spent on data transfer from GPU to CPU, which is now alleviated by reducing the feature dimensions on the GPU. As expected, the tracking time also decreases as K decreases. Figure 5 shows the total average errors (normalized by path length) against the average total processing times for different K, demonstrating that our current approach provides a tradeoff between the processing speed and the accuracy. Instead of using a pre-trained network and a separate PCA step, we believe that training the network with the reduced feature dimension end-to-end will produce a better tradeoff curve in our future work.

4. Conclusion

We presented an algorithm for direct multichannel tracking by using high-dimensional features in a direct image alignment framework for monocular visual odometry. We used the conventional SIFT feature and the more recent CNN features. We employed publicly available datasets to demonstrate that our algorithm, which uses these multichannel features, can provide better pose estimation accuracy than the baseline method using only the intensity (grayscale) channel. Our algorithm is orthogonal to several existing works extending the semi-dense visual odometry [10]: It can be used in a SLAM framework [8] or with a window-based bundle adjustment [7] for better accuracy and globally-consistent reconstruction, and it can work with different camera models such as stereo [9] and omnidirectional cameras [3]. We plan to unify our framework so features can be generated in a way that is optimal for tracking time and accuracy under different circumstances instead of using PCA-based dimensionality reduction heuristics.

Acknowledgments: We thank the anonymous reviewers, Srikumar Ramalingam, Ming-Yu Liu, Teng-Yok Lee, Esra Cansizoglu & Alan Sullivan for their helpful remarks.

References

[1] H. Alismail, B. Browning, and S. Lucey. Enhancing direct camera tracking with dense feature descriptors. In *Proc. Asian Conf. Computer Vision (ACCV)*, 2016. 2, 4, 6

[2] H. Alismail, M. Kaess, B. Browning, and S. Lucey. Direct Visual Odometry in Low Light Using Binary Descriptors. *IEEE Robotics and Automation Letters*, 2(2):444–451, 2017. 2, 4

[3] D. Caruso, J. Engel, and D. Cremers. Large-scale direct SLAM for omnidirectional cameras. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 141–148, 2015. 8

[4] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang. A deep visual correspondence embedding model for stereo matching costs. In *Proc. IEEE Int'l Conf. Computer Vision (ICCV)*, pages 972–980, 2015. 2, 3

[5] A. Crivellaro and V. Lepetit. Robust 3D tracking with descriptor fields. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 3414–3421, 2014. 2

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1052–1067, June 2007. 1

[7] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *arXiv preprint arXiv:1607.02565*, 2016. 2, 8

[8] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Proc. European Conf. Computer Vision (ECCV)*, Sept. 2014. 1, 2, 3, 4, 5, 7, 8

[9] J. Engel, J. Stückler, and D. Cremers. Large-scale direct SLAM with stereo cameras. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 1935–1942, 2015. 4, 8

[10] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *Proc. IEEE Int'l Conf. Computer Vision (ICCV)*, pages 1449–1456, 2013. 1, 2, 3, 4, 8

[11] J. Engel, V. Usenko, and D. Cremers. A photometrically calibrated benchmark for monocular visual odometry. *arXiv preprint arXiv:1607.02555*, 2016. 1, 5

[12] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2014. 1

[13] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 4, 5

[14] W. N. Greene, K. Ok, P. Lommel, and N. Roy. Multi-level mapping: Real-time dense monocular SLAM. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2016. 2

[15] F. Güney and A. Geiger. Deep discrete flow. In *Proc. Asian Conf. Computer Vision (ACCV)*, 2016. 2, 3

[16] A. Handa, T. Whelan, J. B. McDonald, and A. J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, May 2014. 1, 4, 5

[17] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for RGB-D cameras. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, May 2013. 4

[18] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. IEEE Int'l Symp. Mixed and Augmented Reality (ISMAR)*, pages 1–10, Nov. 2007. 1

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 1, 2, 3

[20] C. Liu, J. Yuen, and A. Torralba. SIFT flow: Dense correspondence across scenes and its applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):978–994, 2011. 2

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int'l J. Computer Vision*, 60(2):91–110, 2004. 1, 2

[22] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI)*, pages 674–679, 1981. 2

[23] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5695–5703, 2016. 2, 3

[24] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart. Get out of my lab: Large-scale, real-time visual-inertial localization. In *Proc. Robotics: Science and Systems (RSS)*, 2015. 1

[25] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robotics*, 31(5):1147–1163, 2015. 1

[26] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proc. IEEE Int'l Conf. Computer Vision (ICCV)*, pages 2320–2327, Nov. 2011. 1

[27] S. Park, T. Schöps, and M. Pollefeys. Illumination Change Robustness in Direct Visual SLAM. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2017. 2

[28] M. Peris, A. Maki, S. Martull, Y. Ohkawa, and K. Fukui. Towards a simulation driven stereo vision system. In *Proc. IEEE International Conf. Pattern Recognition (ICPR)*, pages 1038–1042, 2012. 1, 5

[29] H. Strasdat, J. Montiel, and A. J. Davison. Scale drift-aware large scale monocular slam. *Proc. Robotics: Science and Systems (RSS)*, 2010. 5

[30] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 573–580, Oct. 2012. 2, 4

[31] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380, Apr. 1991. 7

[32] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *Proc. European Conf. Computer Vision (ECCV)*, pages 151–158, 1994. 2

[33] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 4353–4361, 2015. 1, 2, 3

[34] J. Zbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2, 2016. 2, 3

[35] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proc. European Conf. Computer Vision (ECCV)*, pages 818–833, 2014. 3